IBM WebSphere® Enterprise Service Bus V6.1 – Lab exercise

WebSphere Enterprise Service Bus lab 3: Use the visual debugger

What this exercise is about

The objective of this lab is to learn how to prepare an environment for debugging a mediation flow, run the debugger and see what happens to the message as it passes through the mediation flow.

Lab requirements

List of system and software required for the student to complete the lab.

- WebSphere Integration Developer V6.1 with the WebSphere Enterprise Service Bus test server option installed.
- This lab is unable to be completed successfully on a z/OS remote machine. Use the integrated test client within WebSphere Integration Developer V6.1.

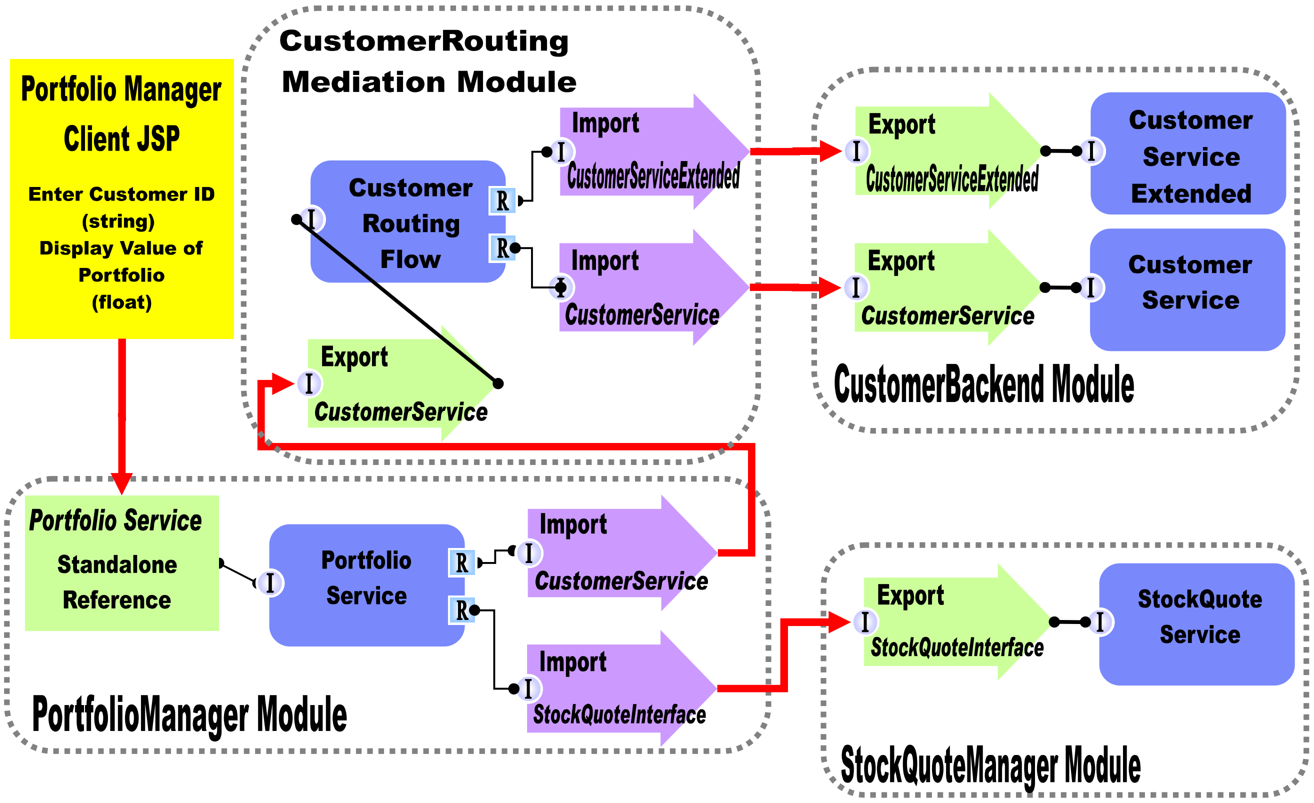
What you should be able to do

At the end of this lab you should be able to:

- Import project interchange files into the WebSphere Integration Developer V6.1 development environment.
- Debug mediation primitives and understand what is happening as a message passes through a mediation flow.
- Set Breakpoints on mediation flow components.

Introduction

In Lab-3 of this lab series, you are going to use the Visual Debugger to step through the above application in order to see what is happening to the message as it passes through the mediation flow. You will first import the application from Lab2 or start with a Project Interchange file that has a working application set up. You will first start server in debug mode. Then, you will open the mediation flow and set breakpoints on all the mediation flow components. Once you have the environment set up, you will bring up a Web browser and use a JSP to send a customerID to the new backend. After submitting the customerID in the JSP, the debug perspective will appear and allow you to step through the application stopping at every breakpoint. This is useful since you can look at the Service Message Object (SMO) in Variables view and see what is changing. Purple check-marks on the wire will show you which components were successfully passed. The message will traverse through the Message Logger, Custom Mediation, Database Lookup, Message Filter, and XSLTransformation in order to show you what each component does to or reads from the SMO information, for the response and request side of the flow.


Portfolio Manager Client JSP
Enter Customer ID (string)
Display Value of Portfolio (float)
CustomerRouting Mediation Module
Customer Routing Flow
Import CustomerServiceExtended
Import CustomerService
Export CustomerService
Export CustomerServiceExtended
Customer Service Extended
Export CustomerService
Customer Service
CustomerBackend Module
Portfolio Service Standalone Reference
Portfolio Service
Import CustomerService
Import StockQuoteInterface
PortfolioManager Module
Export StockQuoteInterface
StockQuote Service
StockQuoteManager Module

Exercise instructions

Some instructions in this lab are Windows operating system specific. If you plan on running WebSphere Integration Developer on a Linux operating system you will need to run the appropriate commands and use appropriate files for Linux. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference Variable	Windows Location	AIX®/UNIX® Location
<WID_HOME>	C:\Program Files\IBM\WID61	/opt/IBM/WID61
<ESB_PROFILE_HOME>	<WID_HOME>\pf\esb	<WID_HOME>/pf/esb
<LAB_FILES>	C:\Labfiles61\WESB\Lab3\import	/tmp/Labfiles61/WESB/Lab3/import
<WORKSPACE>	C:\Labfiles61\WESB\Lab3\workspace	/tmp/Labfiles61/WESB/Lab2/workspace

Windows users' note: When directory locations are passed as parameters to a Java™ program such as EJBdeploy or wsadmin, it is necessary to replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles61\ is replaced by C:/LabFiles61/

Note that the previous table is relative to where you are running WebSphere Integration Developer. The following table is related to where you are running the remote test environment:

Reference variable	Example: Remote Windows test server location	Example: Remote z/OS® test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	sssr011	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/sscell/AppServer	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<SOAP_PORT>	8880	8880	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	ssadmin	
<PASSWORD>	N/A	fr1day	

Instructions for using a remote testing environment, such as z/OS, AIX® or Solaris, can be found at the end of this document, in the section '**Task: Adding remote server to the WebSphere Integration Developer test environment**'.

Part 1: Prepare environment for lab 3

In this section of the lab, you will import all the projects inside the Lab-3 project interchange file into your workspace.

____ 1. Start WebSphere Integration Developer V6.1 with a workspace location of **C:\LabFiles61\WESB\Lab3\workspace**.

____ 2. On the Welcome window, click the curved arrow at top right to **go to workbench**.

____ 3. Import Project Interchange file, **WPIv61_ESB_StartLab3_PI.zip**, into a new workspace.

__ a. Right-click inside **Business Integration View** (top left view in the Business Integration Perspective)

__ b. Select **Import** from the pop-up menu.

__ c. Select **Project Interchange** listed from the Import dialog. Click **Next**.

__ d. Click the top **Browse** button for **From zip file**:

__ e. Navigate to **<LAB_FILES>/WPIv61_ESB_StartLab3_PI.zip**.

__ f. Click the **Select All** button to select all check boxes for projects listed. The projects that must be listed are **CustomerBackend**, **CustomerRoutingMediationModule**, **PortfolioLibrary**, **PortfolioManager**, **PortfolioManagerClient** and **StockQuoteManager**.

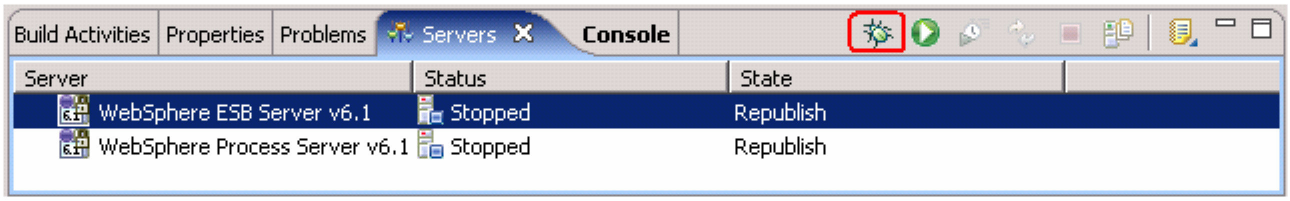
__ g. Verify you have **CustomerBackend**, **CustomerRoutingMediationModule**, **PortfolioLibrary**, **PortfolioManager**, **PortfolioManagerClient** and **StockQuoteManager** modules listed in the Business Integration view.

__ h. Click **Finish** (projects are imported and auto-build will run).

____ 4. Start the ESB Server in debug mode.

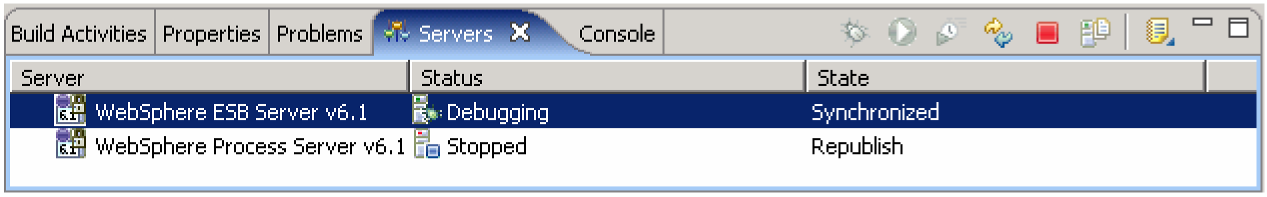
__ a. Verify you have WebSphere ESB Server V6.1 listed in your Servers view.

__ b. Right click on the ESB Server and select Debug or select the ESB server and click the Debug icon ( ). This will start the server in debugging mode.



Server	Status	State
WebSphere ESB Server v6.1	Stopped	Republish
WebSphere Process Server v6.1	Stopped	Republish

__ c. The server starts in a debug mode as shown below:



Server	Status	State
WebSphere ESB Server v6.1	Debugging	Synchronized
WebSphere Process Server v6.1	Stopped	Republish

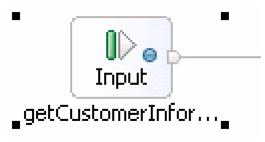
____ 5. While the server is starting in debug mode, add breakpoints to the mediation components. You are setting breakpoints on the mediation components to allow the debugger to stop at each component.

__ a. Open the CustomerRoutingMediationModule **mediation flow**.

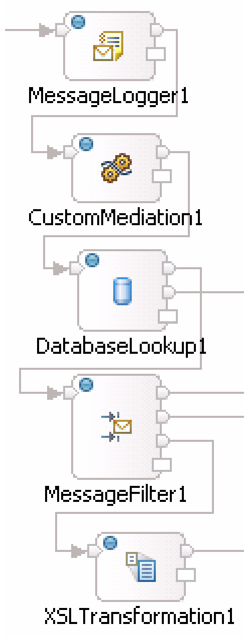
__ b. In the Business Integration view, navigate to **CustomerRoutingMediationModule → Mediation Logic → Flows** and double-click **CustomerRoutingFlow**.

__ c. Click on the **wire** that is connecting CustomerService with CustomerServicePartner and CustomerServiceExtendedPartner. The Mediation Flow will display for the Request side.

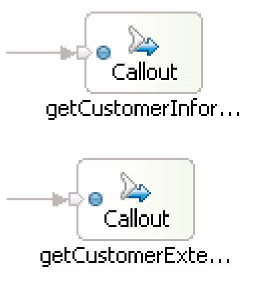
__ d. Right-click on the input (getCustomerInformation: CustomerService, Input node) and select **Debug → Add Breakpoint** from the pop-up menu. Notice the little **blue** icon that appears on the component once a breakpoint has been added.



__ e. Add a breakpoint to all the **mediation primitive components** in the mediation flow; that is, MessageLogger1, CustomMediation1, DatabaseLookup1, MessageFilter1 and XSLTransformation1. Right click each of the components and select **Debug → Add Breakpoint** from the pop-up menu.

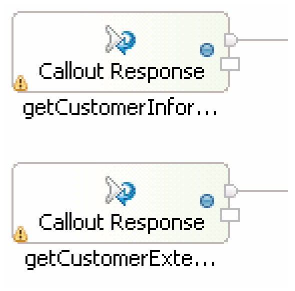


__ f. Add a breakpoint to the **two callout nodes**; that is, the CustomerServicePartner and CustomerServiceExtendedPartner. Right click each of the components and select **Debug → Add Breakpoint** from the pop-up menu.

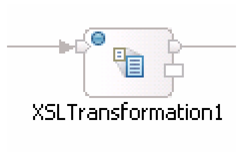


__ g. Now Select the **Response tab** at the bottom of the mediation flow.

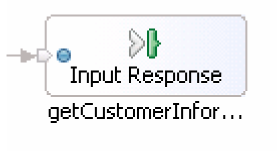
__ h. Add a breakpoint to the two **Callout Response nodes**; that is, the CustomerServicePartner and CustomerServiceExtendedPartner. Right click each of the components and select **Debug → Add Breakpoint** from the pop-up menu.



__ i. Add a breakpoint to the **XSL Transformation1**.



__ j. Add a breakpoint to the **Input Response node**; that is, the CustomerService.



____ 6. Add **projects** to ESB Server and ensure the ESB Server is started in debug mode at this time.

__ a. In Servers view, right-click on '**WebSphere ESB Server V6.1**' and select '**Add and Remove Projects...**'

Note: Notice you are using an ESB profile and not a different server. Therefore, if you have projects with the same name deployed to the server, there may be some naming conflicts. Open the administrative console and stop/uninstall those same-named projects before adding these projects to avoid errors.

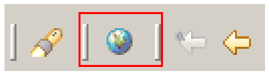
__ b. Click **Add-All** button to move all projects to server and click **Finish** button. Wait for the deployment to finish.

Part 2: Debug mediation flow

In this section you will experience a step by step approach through all the components from the mediation flow in the debug perspective.

____ 1. Start debugging session by submitting a customerID through the JSP.

1) Click on the Web browser icon located on the tools panel, to launch an embedded browser in WebSphere Integration Developer.



2) Enter http://hostname:port/PortfolioManagerClient/index.jsp . Where hostname is the name of the system where the WebSphere Enterprise Service Bus server is located. Port is the **WC_defaulthost** port of the WESB profile.

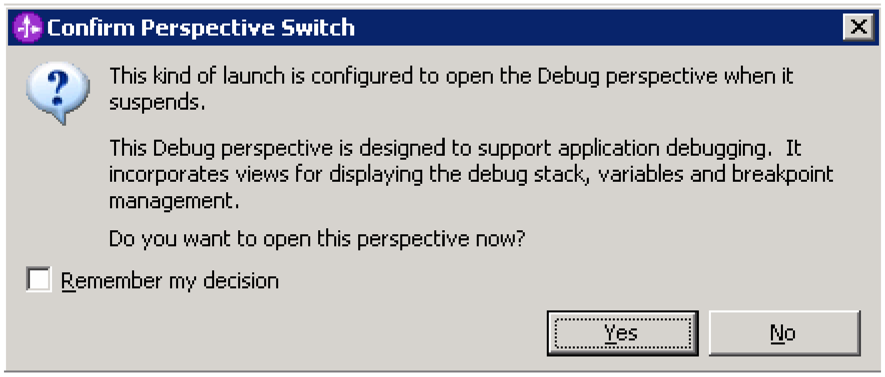
Ex: http://localhost:9080/PortfolioManagerClient/index.jsp

Note: You can get the **WC_defaulthost** port by going to **serverindex.xml** file in <ESB_PROFILE_HOME>\config\cells\esbCell\nodes\esbNode.

3) Enter **7777777** in text input box and click **Submit** to get a response displayed to the JSP and to the Console View of WebSphere Integration Developer.

__ b. Once you have submitted the customerID, the debug perspective will appear and stop at the first breakpoint on the input.

__ c. Click **Yes** when a Confirm Perspective Switch question dialog shows up to switch to the Debug perspective.



Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

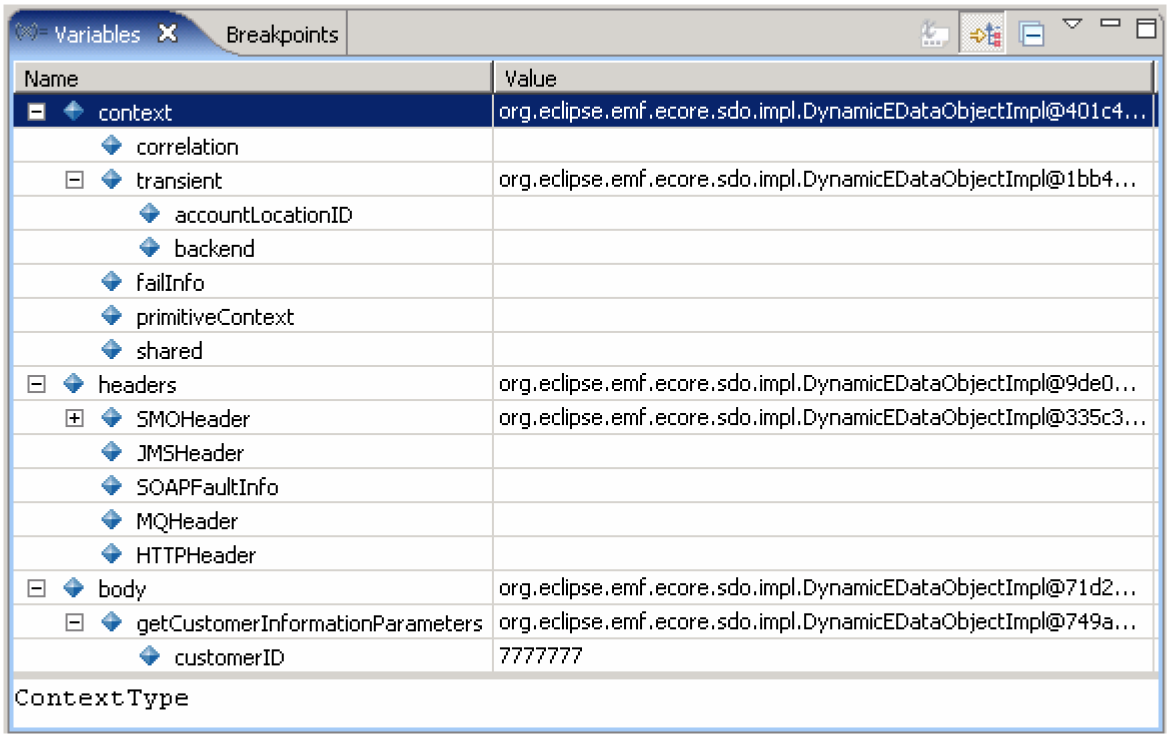
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

☐ Remember my decision

Yes No

____ 2. Now in debug perspective, look at the Variables view (top-right corner) and expand the structure of a Service Message Object (SMO) as shown below:



Name	Value
context	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@401c4...
correlation	
transient	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@1bb4...
accountLocationID	
backend	
failInfo	
primitiveContext	
shared	
headers	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@9de0...
SMOHeader	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@335c3...
JMSHeader	
SOAPFaultInfo	
MQHeader	
HTTPHeader	
body	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@71d2...
getCustomerInformationParameters	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@749a...
customerID	7777777

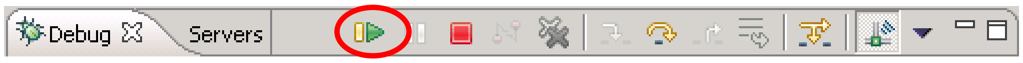
```
ContextType
```

The first section of the SMO holds context information stored in the correlation and transient contexts, and the failInfo. In this lab series, you created a transient context business object that holds an accountLocationID and backend values (both string variables).

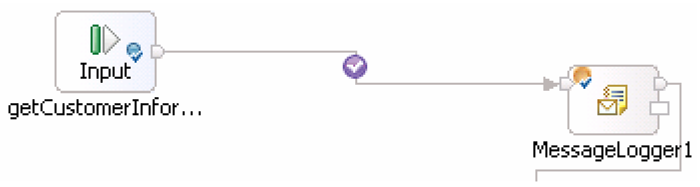
Next is the header information. The SMO has its own header information, along with header information depending upon where the message came from. (Ex: - JMS headers from messaging engines and SOAP headers from a Web service).

Finally there is the body that holds the actual message information. In this case, the message holds the customerID value **7777777**.

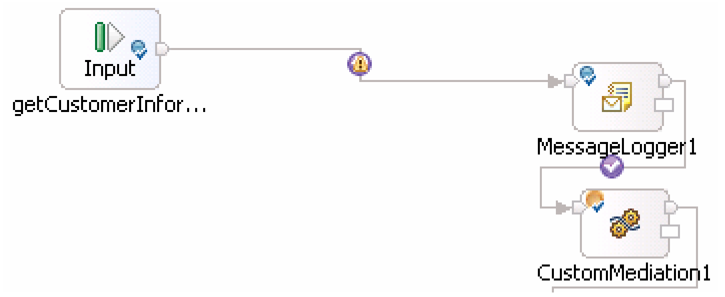
____ 3. Click the **Resume** button in top-left hand side of the debug view ( ).



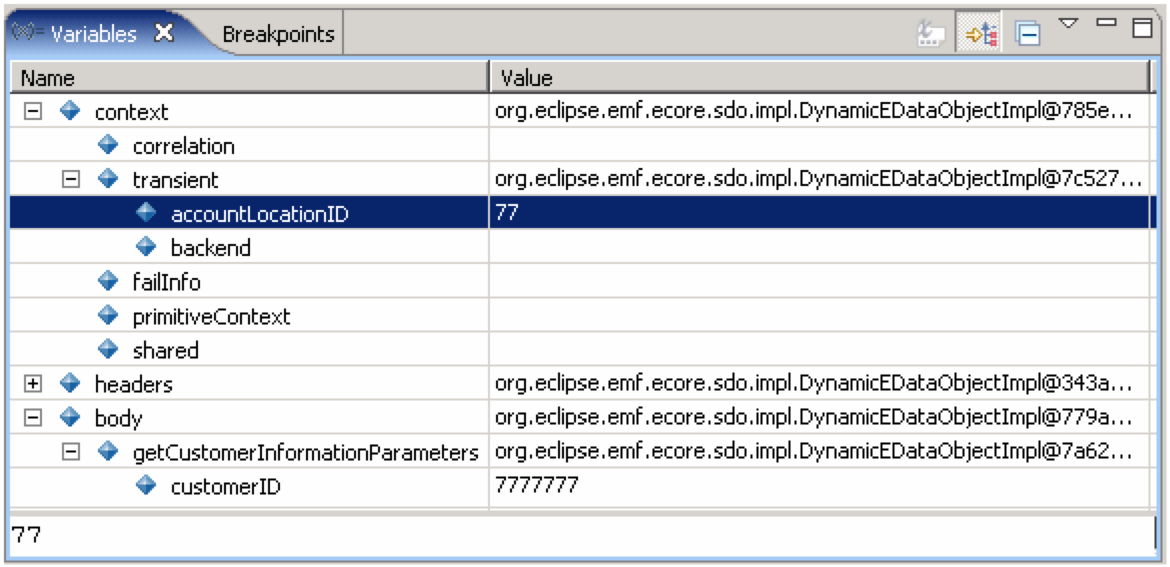
____ 4. In the Mediation Flow Editor view, notice there is a purple check-mark on the wire where the debugger has traversed. In this case, you can see that the debugger has successfully moved the message to MessageLogger1.



____ 5. Click the **Resume** button again. Now the debugger stops at the CustomMediation1 component. The custom mediation uses a Java snippet that will extract a two digit prefix from the customer ID and place 2 digit it in the accountLocationID field of the transient context. In this case, the value will be 77.



____ 6. Click the **Resume** button again. Now that the debugger has traversed through the custom mediation and has reached the DatabaseLookup1, see the transient context in the SMO (Variables View). The accountLocationID should now equal 77 instead of null.



Name	Value
context	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@785e…
correlation	
transient	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@7c527…
accountLocationID	77
backend	
failInfo	
primitiveContext	
shared	
headers	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@343a…
body	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@779a…
getCustomerInformationParameters	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@7a62…
customerID	7777777

77

____ 7. The Database Lookup primitive uses the two digit prefix (77) from the transient context as key to lookup a backend identifier to also place into transient context. The Customer ID prefixes with 11, 22, 33, and 44 will go to the CustomerService backend. Customer ID prefixes with 55, 66, 77, 88, and 99 will go to a CustomerServiceExtended backend. So for this example, the two digit prefix will send the message to the CustomerServiceExtended backend.

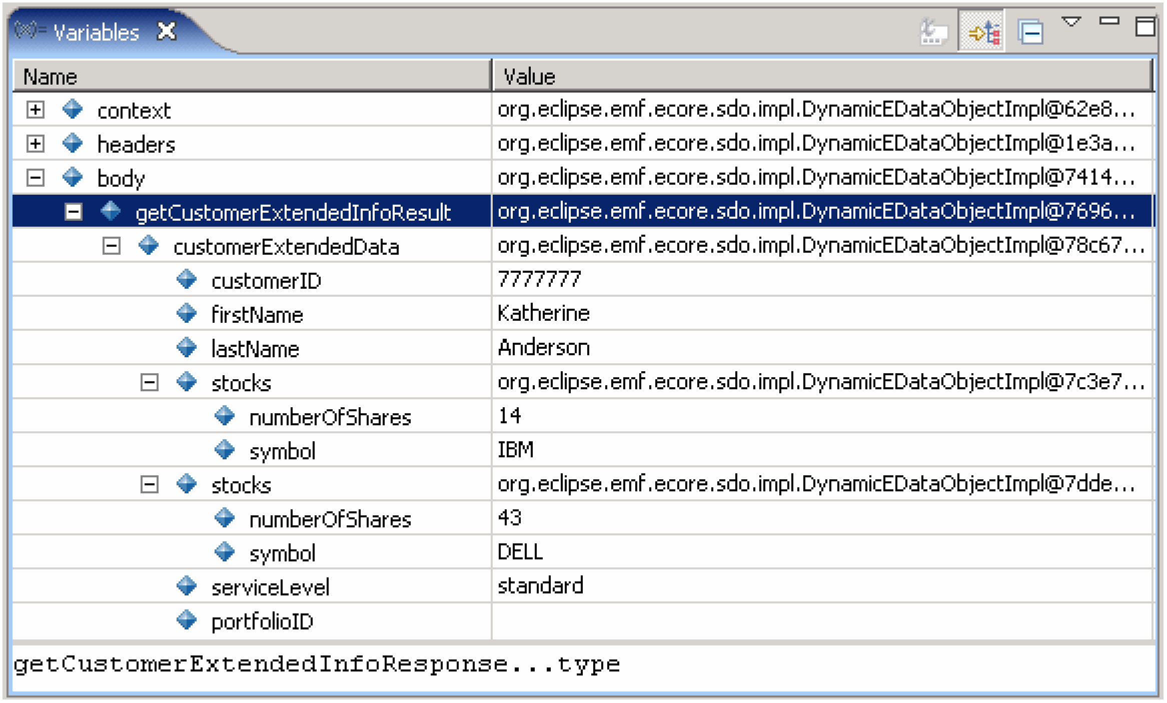
____ 8. Click the **Resume** button again. Check the Variables view to see the transient context in the SMO now has backend = NEW1.

____ 9. Click the **Resume** button again. To determine the message routing based on backend identifier, a Message Filter primitive had been added. The old backend will go directly to the callout for CustomerService and the new backend will go to the XSL Transformation in message filter. In this instance, the message will go to the XSL Transformation.

____ 10. Click the **Resume** button again. Now in the XSL Transformation, check the Variables view to see the SMO body. The XSL Transformation is going the change the body from the Customer business object to the CustomerExtended business object.

____ 11. Click the **Resume** button again. Notice the body has changed to CustomerExtended and has gone to the response side. Click on the **Response: getCustomerInformation**.

____ 12. Click the **Resume** button again. Check the Variables view to see the SMO body has information for a customer on which shares they own and how many shares they have of each company.



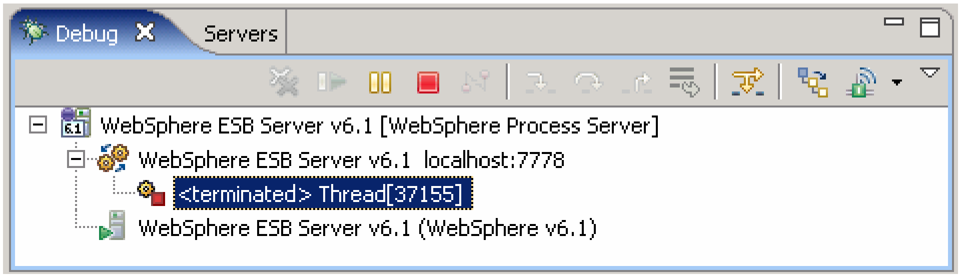
Name	Value
context	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@62e8...
headers	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@1e3a...
body	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@7414...
getCustomerExtendedInfoResult	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@7696...
customerExtendedData	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@78c67...
customerID	7777777
firstName	Katherine
lastName	Anderson
stocks	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@7c3e7...
numberOfShares	14
symbol	IBM
stocks	org.eclipse.emf.ecore.sdo.impl.DynamicEDataObjectImpl@7dde...
numberOfShares	43
symbol	DELL
serviceLevel	standard
portfolioID	

```
getCustomerExtendedInfoResponse...type
```

____ 13. Click the **Resume** button again. Now in the variables view, see how the SMO body has changed. The application only knows how to communicate with the Customer business object and does not know how to use the CustomerExtended business object. Therefore, this XSL Transformation is taking the body in the form of CustomerExtended business object and transforming it back in to a Customer business object so the application understands.

____ 14. You would need to hit resume a couple more times till the process completes.

____ 15. You have now traversed the application to see what happens to the SMO while being operated on mediation primitives. You can see this in the Debug view as shown below if you had traversed through all the break points.



```
WebSphere ESB Server v6.1 [WebSphere Process Server]
  WebSphere ESB Server v6.1  localhost:7778
    <terminated> Thread[37155]
  WebSphere ESB Server v6.1 (WebSphere v6.1)
```

Part 3: Save work and clean up server

____ 1. Export project as Project Interchange file

__ a. Navigate to **File → Export**.

__ b. Select **Project Interchange**.

__ c. Out of all the projects listed, you only need to add a check next to **6 projects**.

CustomerBackend
CustomerRoutingMediationModule
PortfolioLibrary
PortfolioManager
PortfolioManagerClient
StockQuoteManager

All other projects are generated upon import of the project interchange into a workspace.

__ d. Save in C:\LabFiles61\WESB\Lab3

__ e. Name the project interchange **WPIv61_ESB_FinishedLab3_PI.zip**.

____ 2. Go ahead and clean up the **ESB Server**.

1. Start WebSphere ESB Server V6.1 from the Servers view of the Business Integration perspective.
2. Right-click on WebSphere ESB Server v6.1 (once started) and select '**Add and Remove projects...**'
3. Select Remove-All and click Finish.
4. After remove is done, stop the WebSphere ESB Server V6.1.

What you did in this exercise

In this lab you used the Visual Debugger to step through the above application in order to see what is happening to the message as it passes through the mediation flow. You first imported the application from Lab2 or start with a Project Interchange file that has a working application set up. You first started the server in debug mode. Then, you will open the mediation flow and set breakpoints on all the mediation flow components. Once you had the environment set up, you brought up a Web browser and used a JSP to send a customerID to the new backend. After submitting the customerID in the JSP, the debug perspective appeared and allowed you to step through the application stopping at every breakpoint. This was useful since you were able to look at the Service Message Object (SMO) in Variables view and see what was changing. Purple check-marks on the wire showed you which components were successfully passed. The message traversed through the Message Logger, Custom Mediation, Database Lookup, Message Filter, and XSLTransformation in order to show you what each component does to or reads from the SMO information, for the response and request side of the flow.

Solution instructions

This lab has no **Solution** since the Project Interchange file does not save breakpoint information when exported.

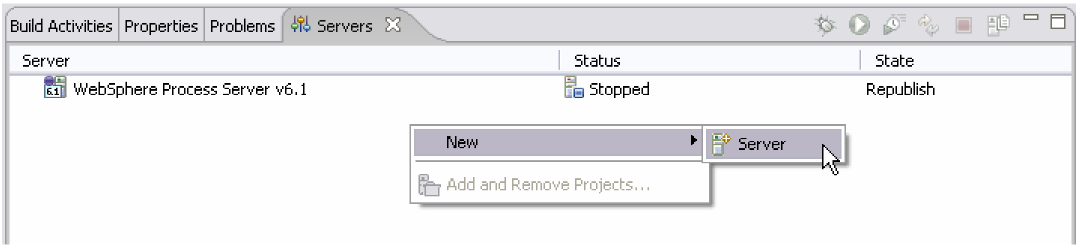
Task: Adding remote server to the WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer test environment. This example uses a z/OS machine.

____ 1. Define a new remote server to WebSphere Integration Developer.

__ a. Right click on the background of the **Servers** view to access the pop-up menu.

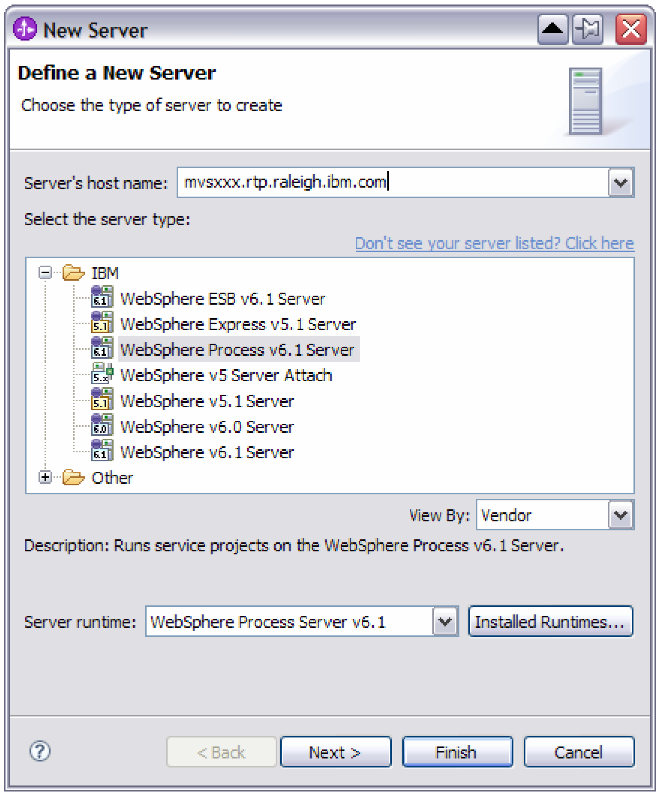
__ b. Select **New → Server**.



Server	Status	State
WebSphere Process Server v6.1	Stopped	Republish

__ c. In the New Server dialog, specify the remote server's host name, **<HOSTNAME>**.

__ d. Ensure that the appropriate server type, '**WebSphere Process v6.1 Server**' or '**WebSphere ESB v6.1 Server**', is highlighted in the server type list



Define a New Server

Choose the type of server to create

Server's host name: mvsxxx.rtp.raleigh.ibm.com

Select the server type:

Don't see your server listed? Click here

- IBM
  - WebSphere ESB v6.1 Server
  - WebSphere Express v5.1 Server
  - WebSphere Process v6.1 Server
  - WebSphere v5 Server Attach
  - WebSphere v5.1 Server
  - WebSphere v6.0 Server
  - WebSphere v6.1 Server
- Other

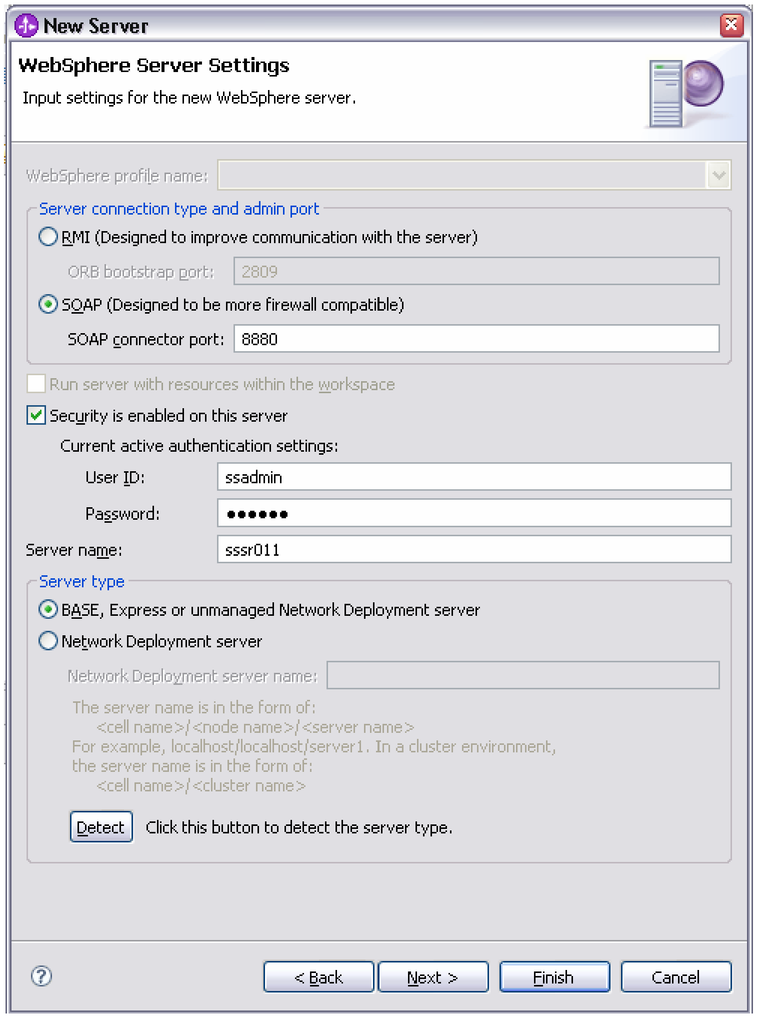
View By: Vendor

Description: Runs service projects on the WebSphere Process v6.1 Server.

Server runtime: WebSphere Process Server v6.1

__ e. Click **Next.**

__ f. On the WebSphere Server Settings page, leave the radio button for **SOAP** selected, changing the **SOAP connector port** to the correct setting (**<SOAP_PORT>**). If security is on in your server, check the box for **'Security is enabled on this server'** and input **<USERID>** for the user ID and **<PASSWORD>** for the password.


New Server
WebSphere Server Settings
Input settings for the new WebSphere server.
Detect
< Back
Next >
Finish
Cancel

WebSphere profile name:

Server connection type and admin port

( ) RMI (Designed to improve communication with the server)

ORB bootstrap port: 2809

(●) SOAP (Designed to be more firewall compatible)

SOAP connector port: 8880

☐ Run server with resources within the workspace

☑ Security is enabled on this server

Current active authentication settings:

User ID: ssadmin

Password: ●●●●●●

Server name: sssr011

Server type

(●) BASE, Express or unmanaged Network Deployment server

( ) Network Deployment server

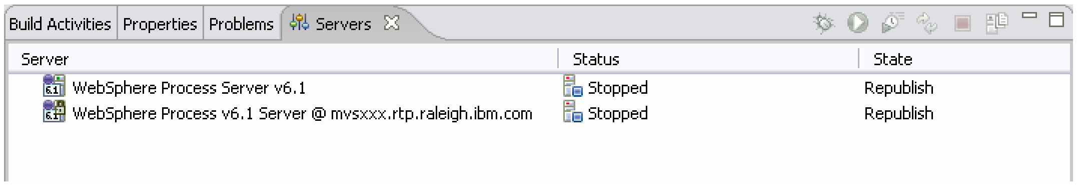
Network Deployment server name:

The server name is in the form of:
<cell name>/<node name>/<server name>
For example, localhost/localhost/server1. In a cluster environment, the server name is in the form of:
<cell name>/<cluster name>

Click this button to detect the server type.

__ g. Click **Finish**.

__ h. The new server should be seen in the Server view.



Server	Status	State
WebSphere Process Server v6.1	Stopped	Republish
WebSphere Process v6.1 Server @ mvsxxx.rtp.raleigh.ibm.com	Stopped	Republish

____ 2. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.

__ a. From a command prompt, telnet to the remote system if needed:

'**telnet <HOSTNAME> <TELNET_PORT>**'

User ID : **<USERID>**

Password : **<PASSWORD>**

__ b. Navigate to the bin directory for the profile being used:

cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin

__ c. Run the command file to start the server: **./startServer.sh <SERVER_NAME>**

__ d. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status
ADMU3000I: Server sssr01 open for e-business; process id is 0000012000000002
```